DISKSCAN USER'S GUIDE

by David Young

INTRODUCTION

DISKSCAN is a powerful and flexible disk utility made to run with the ATARI 810 disk drive. It is designed to include many features for advanced applications and yet be easy enough for a beginner to use.

In a nutshell, DISKSCAN turns the computer's screen into a window looking directly onto the floppy disk. It transforms the mysterious bit patterns of the disk into whatever format is convenient for the application, be it HEX data, ASCII characters or even disassembled object code! While viewing a sector, it can be altered by positioning the cursor over the byte to be changed and then typing the change. Alternatively, a line assembler can be used to modify 6502 object files. A special directory feature reveals the starting sector of any file while a search function can be used to search any file, or even the whole disk, for a 1 or 2 byte sequence. Large groups of sectors can be imaged to another disk, transformed into a binary load file or even dumped to a printer. All of DISKSCAN's functions employ a user interface designed to anticipate responses and detect errors.

Since DISKSCAN is menu driven, users already familiar with the structure of a disk can begin using the program immediately. At some point, however, it would be wise to read the detailed descriptions of the individual commands in order to become privy to their more subtle features. Those desiring to learn more about the disk should begin with the tutorial covering the different types of data structures to be found on the disk. This is not quite as scary as it sounds because there are not that many different types of data and the interactive use of DISKSCAN makes it all quite painless, even fun! Once the use of DISKSCAN is mastered it will become an indispensable tool to any programmer with a disk based ATARI PERSONAL COMPUTER.

SYSTEM REQUIREMENTS

ATARI 800/400 Personal Computer
ATARI 810 Disk Drive
24K RAM (32K for Assembler/Disassembler option)
Printer (optional)

GETTING STARTED

NOTE: Before using this disk, back it up with option J of DOS. However, neither DISKSCAN program will start unless the original distribution disk is in drive 1. Should either program become unusable, make a fresh copy from the backup to the original disk. DO NOT REFORMAT THE ORIGINAL DISK!

1. Turn on your disk drive and insert the DISKSCAN diskette. Insert the BASIC cartridge and power up your computer.

2. When the READY prompt displays, type RUN "DISKSCAN.BIG" for a 32K computer or RUN "DISKSCAN.SML" for a 24K computer.

3. When the DISKSCAN menu appears, remove the DISKSCAN diskette and insert the diskette to be "scanned" into the drive.

THE MENU FUNCTION: M

The first thing that appears on the screen when DISKSCAN is run is the menu. It consists of a list of letters which represent the primary functions available to you along with a short description of each function. In order to call up a function, you simply enter the letter which represents it when asked to do so. If you forget which function certain letters represent, then you can always return to the menu by using the "M" function. Since the function letters are supplied each time a function is requested, you will not find it necessary to return to the menu very often once you have gained a familiarity with DISKSCAN.

Associated with the menu are three questions. Your answers to these questions let DISKSCAN know how the sectors of a file are related to each other (either sequentially or linked), how you would like the sector data to be represented on the screen (either in ATASCII characters or hexadecimal format), and in which drive the disk being scanned is. To change the sector mode, the screen format or the drive #, you can return to the menu with the "M" function. It is also possible (and more convenient) to change the screen format with the "T" (toggle screen format) function. If you are unsure how to answer these questions, it is highly recomended that you read the tutorial on disk data structures before proceeding.

THE FUNCTION PROMPT

To call up a DISKSCAN function, enter the letter which represents that function when the following prompt appears on the screen:

```
R,W,S,C,D,B,I,X,A,T,G,P,H or M?
```

This is referred to as the "function prompt". It consists of a list of the available functions followed by a question mark. Type the letter of the desired function but DO NOT follow it by RETURN. At this point a sector number may be requested. If so, supply the sector number either in decimal or hex (e.g., 20 or $14) and hit RETURN. The function will then be executed. If you cannot remember which letter corresponds to a certain function, use "M" to return to the menu.

INPUT FORMATS

In the interest of user friendliness, DISKSCAN will accept input in four different formats: decimal, hexadecimal, character, and assembly language. Decimal and hex are valid anywhere a number is expected. Characters may be entered if DISKSCAN is in the character mode and you are using either the change ("C") or scan ("S") functions. Assembly language is used in conjunction with the assembler function.

To input a hex number, precede it with "$". Thus, in response to "Sector #?", you could enter either "$100" or "256" with the same result. The only exception to this rule is in the assembler, where most numerical operands MUST be specified in hex but need not be preceeded by "$". The only assembler instructions that must have decimal operands are the conditional branches. The relative displacement MUST be specified as a decimal number preceded by "+" or "-". The reason for these conventions is that the output of the disassembler, which conforms to the same rules, is limited to the right hand margin of the screen. This format actually turns out to be quite convenient for most assembly language applications.

RESPONDING WITH RETURN

When DISKSCAN is awaiting your input, hitting RETURN preceded by no other characters is always a valid response. The program tries to interpret it in a manner convenient to you. For instance, when the function prompt appears, hitting RETURN will cause DISKSCAN to assume the previous function. In response to "Sector #?", a RETURN will cause the program to increment to the next sector if the command was the same, or it will assume the same sector if the command just changed. Within certain contexts, RETURN causes the function to be aborted.

If, for example, you would like to read quickly through several sectors, you would enter "R" in response to the function prompt and then supply the number of the first sector. From then on, if you enter only RETURNs in response to the function and sector prompts, DISKSCAN will continue to read the sectors of a file one after another. This is especially useful if the program is in linked mode as it will automatically calculate the next sector from the link at the end of the current sector.

DIRECTORY OF THE DISK: D

The directory function is useful for determining the location of a file on a DOS disk. (This function is only valid for disks with DOS formats.) It will read the 8 sectors of the directory one at a time and list the filenames of the directory entries along with with the file sizes and, most importantly, the starting sector of each file (all in decimal). It will stop after reading each sector and ask you if you have found the information that you needed. If you answer with anything other than "Y", it will read the next sector of the directory. If you answer with "Y" or if there are no more directory entries, the function prompt will reappear awaiting your next command.

To execute the directory function, type "D" when the function prompt appears. The first sector of the directory will be read and displayed immediately.

READ SECTOR: R

You will probably use this function of DISKSCAN more than any other. It reads an entire sector (128 bytes) from disk and displays its contents either in hex or as ATASCII characters, depending on the current screen format. The 128 bytes are arranged on the screen in a matrix of 8 columns by 16 rows. The address of the the first byte of each row is given in hex in the left margin. The sector number is provided at the top of the screen in both decimal and hex.

To execute the read function type "R" when the function prompt appears. Then provide the sector number either in decimal or hex (preceded by "$") and hit RETURN. The desired sector will be read and displayed immediately. The function prompt will then reappear awaiting your next input.

TOGGLE SCREEN FORMAT: T

This function is useful for changing the format of the screen output from hex to character or vice versa without having to return to the menu. In addition, it actually reprints the screen into the new format at the same time. Thus, if you were using the character format, you could type "T" to observe the sector link in hex and then flip back to character format by typing "T" again.

To execute the screen toggle function, type "T" when the function prompt appears.

PRINT SCREEN CONTENTS: P

This function dumps the contents of the screen to the printer. It does not matter what is being displayed at the time except that, if the screen is in character format, certain characters are unprintable and are printed as dashes. Also note that inverse video characters are printed as normal characters. Thus, you can conveniently make a hard copy of sector data, the menu, or the disk directory. This is especially useful if you are going to alter a sector and you need a record of its original contents.

To execute the screen dump function, type "P" when the function prompt appears.

HEX CONVERSION: H

The hex conversion function is a convenient way to convert hexadecimal numbers to decimal and vice versa. It will operate on any number from 0 to 65535 ($0 to $FFFF). If you use the assembler or disassembler much this function may come in handy.

To execute the hex conversion function, type "H" when the function prompt appears. Then type the number to be converted and hit RETURN. Remember, always precede a hex number by "$". Both the decimal and hexadecimal form of the number will appear just above the function prompt and will remain there until the execution of another function causes it to be erased.

CHANGE CURRENT SECTOR: C

The change sector function is one of the most powerful features of DISKSCAN and yet it is extremely easy to use. A screen editor is implemented, meaning you simply position the cursor over the desired byte of the sector and type the change.

This function is usually preceded by the "R" (read sector) function whereby a sector is read from disk into memory. When you use the change function you are really only altering the image of the sector in memory, not the actual disk sector. The sector on the disk will be updated only if you use the "W" (write sector) function to copy the memory image back to disk.

The cursor is positioned with the normal ATARI cursor control keys: CTRL-, CTRL=, CTRL+, and CTRL* (the little arrows on these keys will not print here). Notice that, when the cursor reaches the end of the line, it will automatically wrap to the beginning of the next line. Likewise, if you try to go below the last line of the sector it will wrap to the top. Thus, since the cursor starts in the top left corner of the sector, the easiest way to get to the end of the sector is to move the cursor to the left. You will probably have to try this out to see how it works. At any rate, it is impossible to move the cursor off of the sector matrix.

When you have the cursor positioned over the byte you desire to change, type the new byte over the old. This will require 1 keystroke per byte when the screen is in character format or 2 keystrokes per byte in the hex format. Notice that the cursor automatically moves over as you type the change. Thus, you can alter any number of successive bytes without having to reposition the cursor. When in hex format, only valid hex characters are allowed. If you try to type any other characters, the sector buffer in memory will not be altered but the buzzer will sound and a blank will be left in that nybble position, indicating you need to go back and correct it. However, when in character format, ALL keystrokes print their ATASCII representations except the cursor controls and ESC, which is used to exit the change function. Of course, BREAK is also not allowed unless you want to suspend execution of DISKSCAN. If you should accidently hit BREAK, clear the screen and type RUN. Your sector will still be in memory.

To execute the change function, type "C" when the function prompt appears. Position the cursor and type the change. Exit the screen editor with ESC. Now write the sector back out to disk with the "W" (write sector) function if you desire to save it (see next section).

WRITE SECTOR TO DISK: W

This function is used to copy the sector image in memory to any sector on disk. Using it in conjunction with the change function ("C") or the assembler ("A") allows you to alter disk data directly. This can be very convenient for recovering an accidently erased file, patching a boot disk and many other operations which cannot easily be performed through DOS. However, certain precautions should be taken to avoid loss of disk data integrity. First of all, always make a backup copy of the disk that you are going to alter. Secondly, it is a good idea to make a hard copy of a sector with the screen dump function ("P") before you alter it.

Because the write function should not be used quite so casually as the other DISKSCAN functions, it is implemented so as to prevent accidental use. Before doing the write operation, the program will ask you if you are sure that you want to output to the specified sector number. You must answer with a "Y" and a RETURN for the write to take place.

One short cut available to you when using this function (as well as others) is that you can usually just hit RETURN when "Sector #?" appears. This is because DISKSCAN will assume the same sector number if the write function was preceded by a different function (typically "C"). When the write function is preceded by itself, if, say, you are writing the same memory image to several sequentially numbered sectors, the program will automatically increment to the next sector.

To execute the write function, type "W" when the function prompt appears and then provide a sector number (optional) followed by RETURN. The current sector in memory will be displayed and you are prompted to confirm your intent to write to the specified sector. A response other than "Y (RETURN)" will safely abort the operation.

SCAN DISK FOR 1 OR 2 BYTES: S

One area where computers really shine is in their ability to quickly search a massive amount of data for a specific sequence. The scan function of DISKSCAN will automatically search a disk for a 1 or 2 byte sequence which you specify. The scan sequence can be specified as numbers if the screen is in hex format or as characters if in character format. Once the search is underway, the program will stop scanning the disk if it encounters the specified sequence, display the sector, and ask you if it has found the right one. Any response other than "Y" will cause the search to resume where it was left off. It will stop scanning only if the end of file is encountered (if in linked mode) or the end of disk is reached (if in sequential mode).

You might now be asking the clever questions: "What happens if the 2 bytes are split between the end of one sector and the beginning of another? And what about the sector links? Are they included in the search?" The answer is: Don't worry. DISKSCAN does the right thing in all cases. If the first byte of the sequence matches the last byte of a sector and the second byte of the sequence matches the first byte of the next sector, the previous sector will be reread and displayed. As to whether the sector links are included in the search, it depends on the sector mode: in sequential mode the links are included, in linked mode they aren't. Analogous situations arise in both the assembler and disassembler functions and they are handled just as expediently.

To execute the scan function, type "S" when the function prompt appears. Then provide the sector number where the scan will begin and hit RETURN. DISKSCAN will read and display the first sector and prompt you for the first byte of the scan sequence. Type a number (not greater than 255 or $FF) or character, which ever is appropriate for the current screen format, and hit RETURN. When prompted for the second byte, either supply it or just hit RETURN if you want to search for a single byte. If you want to specify a control character, be sure to precede it by ESC so that it will print instead of execute. The RETURN character can be specified only with the screen in hex format ($9B). When the search starts, a visible counter ticks off the sector numbers as they are scanned. If the sequence is found, the sector will be displayed with a small arrow pointing to the first byte of the sequence. If you wish the scan to continue, hit any key but "Y". While the scan is in progress, it can be aborted by hitting RETURN.

IMAGE SECTORS: I

This function is used to make an exact copy of one or more sequentially numbered sectors to another disk. The imaged sectors will retain their same numbers on the destination disk. While this function could be used to copy an entire disk, its main purpose is to allow you to easily keep your backup current as you modify a disk. In other words, at convenient points during the modification of a disk you should make images of the sectors you have been working on to a secondary backup disk (not to the original backup). Of course, the read and write functions ("R" and "W") could just as easily be used if you only need to copy 1 or 2 sectors.

The number of sectors you can image at one time is limited by the amount of free memory space available. Thus, the more RAM you have the better. However, there is another way to make more RAM available to the image function. If you have at least 32K of RAM then you will normally want to run DISKSCAN.BIG because it has the assembler/disassembler functions available. DISKSCAN.SML is the shorter implementation of DISKSCAN, made to run out of at least 24K. Thus, even if you have more than 24K of RAM, you may use DISKSCAN.SML to make available to the image function the maximum free memory space. This same discussion applies to the binary load file function ("B") as well.

To execute the image function, type "I" when the function prompt appears. Supply the starting sector number and hit RETURN. You will then be informed of the maximum number of sectors you can image at one time. Supply the number of sectors you wish to image and hit RETURN. When prompted, insert the destination disk and hit RETURN. When the function prompt appears, the image operation is complete and you may reinsert the source disk.

BINARY LOAD FILE: B

This function is used to make a binary load file out of one or more sectors. That is, you can make a DOS file out of the sectors of, say, a game boot disk. Then you can load the file into memory with the binary load option of DOS and perhaps disassemble it or even execute it. It will work in either sector mode, sequential or linked, but it will usually only make sense in the sequential mode. Though this is a very powerful function, it will be useful only to advanced users.

The number of sectors you can make into a binary load file at one time is limited by the amount of free memory space available. Thus, the more RAM you have the better. However, there is another way to make more RAM available to the binary load file function. If you have at least 32K of RAM then you will normally want to run DISKSCAN.BIG because it has the assembler/disassembler functions available. DISKSCAN.SML is the shorter implementation of DISKSCAN, made to run out of at least 24K. Thus, even if you have more than 24K of RAM, you may use DISKSCAN.SML to make available to the binary load file function the maximum free memory space. This same discussion applies to the image sectors function ("I") as well.

To execute the binary load file function, type "B" when the function prompt appears. Supply the starting sector number and hit RETURN. You will then be informed of the maximum number of sectors you can make into a binary load file at one time. Supply the desired number of sectors and hit RETURN. When prompted, insert the destination disk (make sure it is a DOS disk), supply the load address to be appended at the beginning of the file, and hit RETURN. When the function prompt appears, the binary load file is complete and you may reinsert the source disk.

ASSEMBLY LANGUAGE SUPPORT

Working in hex or character format is fine for many DISKSCAN applications, but for inspecting and patching 6502 machine language files, assembly language support is almost a necessity. To this end, DISKSCAN not only offers an assembler and disassembler, but also a unique "GOTO binary address" function for locating specific addresses within a binary file. The assembly language facilities are only included in DISKSCAN.BIG for running on machines with at least 32K of RAM. There are several books on the market which cover 6502 assembly language. One that I recommend is Beyond Games: Systems Software for Your 6502 personal computer by Ken Skier.

DISASSEMBLE A SECTOR: X

The disassembler function can translate the 6502 machine code of a binary file into standard assembly language instructions. It will disassemble, outputting to the right margin of the screen, from the point you specify within a sector until the end of the sector. If you desire, it will continue disassembling with the next logical sector, predicated on the current sector mode: sequential or linked. A multibyte instruction split between 2 sectors is handled gracefully with no discontinuity. Little arrows in the sector data matrix indicate the region of the sector being disassembled.

The output of the disassembler is designed to fit into the limited space to the right of the sector data matrix. For that reason, most numerical operands are specified in hex, but without the preceding "$" which usually distinguishes hexadecimal from decimal. The only disassembler instructions which have decimal operands are the conditional branches and these are always preceded by "+" or "-" to indicate the direction of the relative displacement. To find the destination of the branch, start at the beginning of the next instruction and count forward or backwards the number of bytes indicated by the displacement.

Several other DISKSCAN functions compliment the disassembler. If you need a hard copy of the disassembler output, use the "P" function. A hexadecimal operand can be converted to decimal with the "H" function. If you need to find a specific address within a machine language program, use the "G" function.

To execute the disassembler function, type "X" when the function prompt appears. The current sector will be displayed and you will be prompted for the starting byte. Since the byte addresses are provided in hex to the left of the sector matrix, the starting byte is most easily provided in hex (preceded by "$"). The disassembly will proceed from that point until the right margin is full. When the program asks if that is enough, any response other than "Y" will cause the disassembly to continue. This is also the case when the end of the sector is reached. Answering "Y" to the prompt will cause the disassembler to be terminated and the function prompt to appear.

ASSEMBLE INTO SECTOR: A

When modifying 6502 machine language, a convenient alternative to the change function ("C") is the assembler function. It implements a line assembler (meaning that each instruction is assembled as you type it in) to modify the current sector in memory. It accepts standard 6502 assembly language instructions and will beep at you if you attempt to input something illegal. You can start assembling anywhere within the sector (except within the sector link if in linked mode). When the end of the sector is reached, if desired, the current sector will be written out and the next logical sector (predicated on the current sector mode: sequential or linked) will be read in. A multibyte instruction split between 2 sectors is handled gracefully with no loss of data.

The numerical operands of the assembler conform to the same format as the disassembler. All numerical operands, except the displacements of conditional branches, must be given in hex and may or may not be preceded by "$". The relative displacements of conditional branches must be specified in decimal preceded by "+" or "-" to indicate the direction of the branch. To calculate this displacement, start at the first byte of the following instruction and count forward or backwards the number of bytes to the destination address.

Should any question arise as to how to specify any of the 13 6502 addressing modes, study the output of the disassembler. Leading zeros and imbedded blanks are ignored by the assembler.

To execute the assembler function, type "A" when the function prompt appears. Provide the starting byte (most conveniently in hex preceded by "$") and hit RETURN. You may then begin entering assembly language instructions followed by RETURN. Each instrucion will be translated into machine code and inserted into the sector buffer, overwriting the current contents. A little arrow in the sector matrix indicates the point at which the next instruction will be inserted into the sector. If the end of the sector is reached, you can cause the current sector to be written out and the next logical sector to be read in by responding to the prompt with "Y". Any other response causes the assembler to be terminated. The assembler can also be terminated by hitting RETURN when prompted for the next instruction. Upon termination, the function prompt will reappear.

GOTO BINARY ADDRESS: G

Inspecting a binary file can be very tedious if the flow of the program jumps around much. This is because it is very time consuming to calculate which byte of which sector is referenced by an absolute address. In fact, if the task at hand is a large one, it is recommended that you use the "B" function to create a binary load file out of the sectors of the program (unless, of course, it already is a binary load file with a convenient load address). This would allow you to load the program into memory for processing by a full blown disassembler. However, for casually inspecting a binary file, the GOTO function is a good way to find your way around. It allows you to jump to any absolute address within the file without doing a single calculation.

When using the GOTO function, you can take advantage of the fact that the load address is usually located at the very beginning of a program. After you have specified this as the base address then the GOTO function can calculate the exact location on the disk of any absolute address within that program. It bases its calculation on either 125 bytes/sector, if the sectors are linked, or 128 bytes/sector, if they are sequential. Once the address is found, the disassembler ("X") can be executed starting at that location.

To execute the GOTO function, type "G" when the function prompt appears. Then provide the value, sector, and byte number of the base address as they are requested. If you have already entered these once, just hit RETURN at the first prompt. You will then provide the destination address. When you hit RETURN, the program will go find the address, display the sector and print the byte number of the address above the function prompt so that it can be referenced when choosing the next function. Remember, if the base address remains the same, it is not necessary to specify it after the first use of the GOTO function.

QUESTIONS?

I have tried to make this program as powerful and user friendly as possible. I would appreciate any comments or suggestions. Also, if you have any questions, feel free to write.

David Young
421 Hanbee
Richardson, TX
75080